IN THE CLAIMS:

Please amend the claims as follows. The claims are in the format as required by 35 C.F.R. § 1.121.

1. (Currently amended) A method of modeling an arbitrarily complex environment, comprising:

on a computer having a computer memory and a processor, defining a plurality of types of data structures ~~for dynamically accommodating changes to the arbitrarily complex environment~~ in a data model, wherein each of the data structures comprises ~~components and relationships~~ one or more fields or properties associated with the data structure, wherein all data structures of the same type contain the same properties;

instantiating a component for ~~representing~~ each atomic entity in the arbitrarily complex environment ~~with a component in the data model, wherein each atomic entity is a logical or physical entity in the arbitrarily complex environment and~~ wherein ~~the~~ each component has a set of fields which contain information relating to the atomic entity associated with the component, wherein the set of fields comprises:

a set of property fields containing information about the attributes or characteristics of the component; and

a field that contains a link to its component type;

assigning values to the properties in the instantiated component based on the attributes of the entity which the component was instantiated to represent;

instantiating a relationship for representing an association or a dependency between two or more components in the data model ~~with a relationship~~, wherein **[[**the**]]** each relationship **[[**has**]]** comprises:

~~a name field containing a relationship name which is built programmatically and which automatically associates the relationship with two or more physical or logical entities that correspond to the two or more components~~

a field that is a foreign key to its relationship type; and

a set of property fields containing information about one or more of the attributes of the relationship; and

storing the components in a schema, wherein property definitions of each component are linked to a type of component, wherein changes made to the type of component are automatically associated with all components of that type of component without changing the

schema ~~automatically changing the relationship~~ to reflect a corresponding change in the arbitrarily complex environment.

2. (Currently amended) The method of claim 1, wherein each component is instantiated based on a generic component type and has a set of core attributes comprising an id, a name, a description, a type, and a set of properties ~~property, and the presence of events, wherein the name field associates the component with a particular atomic entity~~.

3. (Previously Presented) The method of claim 1, wherein each component type is in a hierarchy of component types.

4. (Currently amended) The method of claim 2, wherein each property has a data type of one ~~or more~~ of a string, a numeric, a Boolean, a link, a date/time and a custom type.

5. (Previously Presented) The method of claim 2, wherein each property is a data structure having a name, a description and a value.

6. (Cancelled).

7. (Cancelled).

8. (Previously Presented) The method of claim 1, wherein each relationship type is a parent type or a subtype.

9. (Cancelled).

10. (Currently amended) The method of claim **[[9]]** 1, wherein each component is represented in a component table.

11. (Original) The method of claim 10, wherein each component type is represented in ~~component type table~~.

12. (Original) The method of claim 11, wherein each relationship is represented in a relationship table.

13. (Original) The method of claim 12, wherein each relationship type is represented in relationship type table.

14. (Currently amended) The method of claim 13, wherein the relationship table links each relationship to ~~at least~~ exactly two components.

15. (Previously Presented) The method of claim 14, wherein the relationship table and the relationship type table are distinct.

16. (Currently amended) A system for modeling an arbitrarily complex environment, comprising:

a computer having a memory for storing a set of computer-executable instructions and a processor for executing the computer-executable instructions operable to:

define a plurality of data structures ~~for dynamically accommodating changes to the arbitrarily complex environment~~ in a data model, wherein each of the data structures comprises ~~components and relationships~~ one or more fields of properties associated with the data structure, wherein all data structures of the same type contain the same properties;

~~generate~~ instantiate a set of components in the data model, wherein each component in the set of components represents ~~each~~ a particular atomic entity within the arbitrarily complex environment, ~~wherein each atomic entity is a logical or physical entity in the arbitrarily complex environment and~~ wherein each component has a set of fields which contain information relating to the atomic entity associated with each component, wherein the set of fields comprises;

a property field containing information about the attributes or characteristics of the component; and

assign values to the properties in the instantiated component based on the attributes of the entity which the component was instantiated to represent;

~~generate~~ instantiate a set of relationships in the data model, wherein each relationship represents an association between at least two of the components, wherein **[[**the**]]** each relationship **[[**has**]]** comprises:

~~a name field containing a relationship name which is built programmatically and which automatically associates the relationship with two or more physical or logical entities that correspond to the two or more components~~

a property field containing information about one or more of the attributes of the relationship; and

store the components in a schema, wherein property definitions of each component are linked to a type of component, wherein changes made to the type of component are automatically associated with all components of that type of component without changing the schema, and wherein one or more fields in a relationship are changed based on a check in the relationship check field to reflect ~~automatically change the relationship to reflect~~ a corresponding change in the arbitrarily complex environment.

17. (Currently amended) The system of claim 16, wherein each component is instantiated based on a generic component type and has a set of core attributes comprising an id, a name, a description, a type, and a set of properties ~~property, and the presence of events, wherein the name field associates the component with a particular atomic entity~~.

18. (Currently amended) The system of claim 16 **[[ ]]**, wherein each component type is in a hierarchy of component types.

19. (Currently amended) The system of claim 18, wherein each property has a data type of one ~~or more~~ of a string, a numeric, a Boolean, a link, a date/time and a custom type.

20. (Previously Presented) The system of claim 19, wherein each property is a data structure having a name, a description and a value.

21. (Cancelled).

22. (Cancelled).

23. (Currently amended) The system of claim **[[22]]** 16, wherein each relationship type is a parent type or a subtype.

24. (Cancelled).

25. (Currently amended) The system of claim **[[21]]** 16, wherein each component is represented in a component table.

26. (Original) The system of claim 25, wherein each component type is represented in component type table.

27. (Original) The system of claim 26, wherein each relationship is represented in a relationship table.

28. (Original) The system of claim 27, wherein each relationship type is represented in relationship type table.

29. (Currently amended) The system of claim 28, wherein the relationship table links each relationship to ~~at least~~ exactly two components.

30. (Previously Presented) The system of claim 29, wherein the relationship table and the relationship type table are distinct.

31. (Currently amended) A software product ~~capable of~~ comprising a set of instructions stored ~~instructing a computer~~ on a computer-readable medium, wherein the computer has a computer memory and a processor for executing the set of instructions, wherein the software product comprises:

an instruction to define a plurality of data structures ~~for dynamically accommodating changes to an arbitrarily complex environment~~ in a data model, wherein each of the data structures comprises ~~components and relationships~~ one or more fields of properties associated with the data structure, wherein all data structures of the same type contain the same properties;

an instruction to ~~represent~~ instantiate a component for each atomic entity in the arbitrarily complex environment, ~~wherein each atomic entity is a logical or physical entity in the arbitrarily complex environment and~~ wherein ~~the~~ each component has a set of fields which contain information relating to the atomic entity associated with the component, wherein the set of fields comprises:

a property field containing information about the attributes or characteristics of the component; and

an instruction to assign values to the properties in the instantiated component based on the attributes of the entity which the component was instantiated to represent;

an instruction to ~~represent~~ instantiate a relationship to represent an association or a dependency between two or more components ~~with a relationship~~, wherein **[[**the**]]** each relationship **[[**has**]]** comprises:

a property field containing information about one or more of the attributes of the relationship ~~a name field containing a relationship name which is built programmatically and which automatically associates the relationship with two or more physical or logical entities that correspond to the two or more components relationships~~; and

an instruction to store the components in a schema, wherein property definitions of each component are linked to a type of component, wherein changes made to the type of component are automatically associated with all components of that type of component without changing the schema, wherein one or more fields in a relationship are changed to reflect ~~automatically change the relationship to reflect~~ a corresponding change in the arbitrarily complex environment.

32. (Currently amended) The software product of claim 31, wherein each component is instantiated based on a generic component type and has a set of core attributes comprising an

id, a name, a description, a type, and a set of properties ~~property, and the presence of events, wherein the name field associates the component with a particular atomic entity~~.

33. (Previously Presented) The software product of claim 31, wherein each component type is in a hierarchy of component types.

34. (Currently amended) The software product of claim 33, wherein each property has a data type of one ~~or more~~ of a string, a numeric, a Boolean, a link, a date/time and a custom type.

35. (Previously Presented) The software product of claim 34, wherein each property is a data structure having a name, a description and a value.

36. (Cancelled).

37. (Cancelled).

38. (Previously Presented) The software product of claim 37, wherein each relationship type is a parent type or a subtype.

39. (Cancelled).

40. (Currently amended) The software product of claim [[6]] 31, wherein each component is represented in a component table.

41. (Previously Presented) The software product of claim 40, wherein each component type is represented in component type table.

42. (Previously Presented) The software product of claim 41, wherein each relationship is represented in a relationship table.

43. (Previously Presented) The software product of claim 42, wherein each relationship type is represented in relationship type table.

44. (Currently amended) The software product of claim 43, wherein the relationship table links each relationship to ~~at least~~ exactly two components.

45. (Previously Presented) The software product of claim 44, wherein the relationship table and the relationship type table are distinct.

46. (Currently amended) The method of claim 1, further comprising, utilizing a typing system to define the hierarchy of components and relationships.

47. (Previously Amended) The method of claim 46, wherein the typing system further includes a generic model structure to define a hierarchy of components and relationships.

48. (Previously Amended) The method of claim 47, wherein a data structure is associated with the generic data model.

49. (Previously Presented) The method of claim 48, wherein the data structure associated with the generic data model is stored utilizing a table schema.

50. (Previously Presented) The method of claim 49, wherein the table schema does not change with an addition of a data structure or types of data structures.

51-53. (Cancelled).